PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-201791
(43)Date of publication of application : 22.07.1994

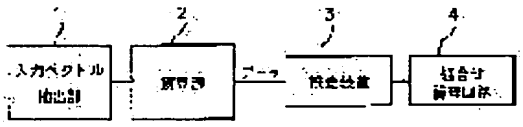
(51)Int.Cl. G01R 31/28
G06F 11/22

(21)Application number : 04-360139
(22)Date of filing : 28.12.1992

(71)Applicant : CANON INC
(72)Inventor : FUKATSU TSUTOMU

(54) GENERATING DEVICE FOR INSPECTION DATA OF COMBINATORIAL LOGIC CIRCUIT

(57)Abstract:
PURPOSE: To provide a combinatorial logic circuit inspection data generating device being simple in operation process.
CONSTITUTION: An operation section 2 defines circuit formation and fixed logical expression, and an input vector extracting section 1 selects such a first test vector that the failure state of a detectable node becomes maximum. After the failure state of the node detected by the test vector is removed, a second test vector is further extracted so that number of the failure states of detectable nodes becomes maximum and until the whole failure states of the all nodes are detected, after the failure states of the nodes detected by the test vectors from the first to the (n-1)th are removed, a process for extracting the (n)th test vector where the number of the failure states of the detectable nodes becomes maximum is further repeated to generate test data where the failure states of the whole nodes are detected, and the test data are supplied to an inspecting device 3.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平6－201791

(43)公開日　平成６年(1994)７月22日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０１Ｒ　31/28					
Ｇ０６Ｆ　11/22	３１０　Ｂ	7737－5Ｂ			
		6912－2Ｇ	Ｇ０１Ｒ　31／28	Ｑ	

審査請求　未請求　請求項の数３　（全 18 頁）

(21)出願番号　特願平4－360139

(22)出願日　平成４年(1992)12月28日

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号

(72)発明者　普勝　勉
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内

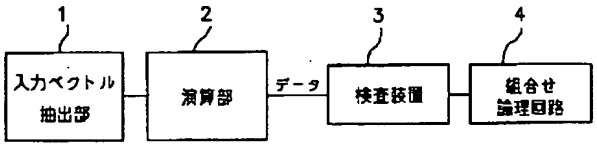
(74)代理人　弁理士　國分　孝悦

(54)【発明の名称】　組合せ論理回路検査データ発生装置

(57)【要約】

【目的】　演算処理の簡単な組合せ論理回路検査データ発生装置を得る。

【構成】　演算部２で回路構成及び所定の論理式を定義し、入力ベクトル抽出部１により、①検出可能なノードの故障状態が最大になるような第１のテストベクトルを選出し、②上記テストベクトルにより検出されるノードの故障状態を除いた上で、更に検出可能なノードの故障状態数が最大となる第２のテストベクトルを抽出し、全ノードの全故障状態が検出されるまで、第１から第ｎ－１までのテストベクトルにより検出されるノードの故障状態を除いた上で、更に検出可能なノードの故障状態数が最大となる第ｎのテストベクトルを抽出する動作②を繰り返して、全ノードの故障状態を検出するようなテストデータを発生させ、検査装置３に供給する。



1

【特許請求の範囲】

【請求項1】　複数の論理素子により構成された組合せ論理回路に対する全入力の組合せを試行し、故障検出数が最大となる第1の入力ベクトルを抽出し、更に上記第1の入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる第2の入力ベクトルを抽出する試行を行い、第n－1，n－2，…，2，1の入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる第nの入力ベクトルを抽出することにより得られたデータを、上記組合せ論理回路の回路構成が所望の通りになっているか否かを検査する検査装置に供給する組合せ論理回路検査データ発生装置において、

上記回路構成の定義を上記論理素子各々の入出力関係について行い、その際の順序づけを、各論理素子の入出力点をa，…，h，iとしたとき、

ac(j)=fac(ac(i),ac(h),…,ac(a))　ただし、j⟩a,…,h,i

なる関係を有するように定義付けられた論理式を用いてデータの発生を行う演算手段を設けたことを特徴とする組合せ論理回路検査データ発生装置。

【請求項2】　複数の論理素子により構成された組合せ論理回路に対する全入力の組合せを試行し、故障検出数が最大となる第1の入力ベクトルを抽出し、更に上記第1の入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる第2の入力ベクトルを抽出する試行を行い、第n－1，n－2，…，2，1の入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる第nの入力ベクトルを抽出することにより得られたデータを、上記組合せ論理回路の回路構成が所望の通りになっているか否かを検査する検査装置に供給する組合せ論理回路検査データ発生装置において、

上記回路構成の定義を上記論理素子各々の入出力関係について行い、その際の順序づけを、各論理素子の入出力点をa，…，h，iとしたとき、

ac(j)=fac(ac(i),ac(h),…,ac(a))　ただし、j⟩a,…,h,i

なる関係を有するように定義付けられた論理式を用いてデータの発生を行い、

st(j)=st(k) & fst(ac(i),ac(h),…,ac(a))　ただし、k⟩j,i,h,…,a

なる関係を有するように定義付けられた論理式を用いて条件の評価を行う演算手段を設けたことを特徴とする組合せ論理回路検査データ発生装置。

【請求項3】　複数の論理素子により構成された組合せ論理回路に対する全入力の組合せを試行し、故障検出数が最大となる第1の入力ベクトルを抽出し、更に上記第1の入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる第2の入力ベクトルを抽出する試行を行い、第n－1，n－2，…，2，1の入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる第nの入力ベクトルを抽出することにより得られたデータを、上記組合せ論理回路の回路構成が所望の通りになっているか否かを検査する検査装置に供給する組合せ論理回路検査データ発生装置において、

2

上記回路構成の定義を上記論理素子各々の入出力関係について行い、各論理素子の入出力点としてのノードa，b，…，h，iにノード情報の記憶領域を有し、入力端子に接続されたノードを初期化したのち、定義された論理演算の実行順として、上記ノードに論理素子を介して接続されているノード群のノード情報が全て確定しているとき、上記ノード群のノード情報の中で最大のノード情報を単調増加させた値をそのノードのノード情報として記憶し、上記ノード情報が全ノードで確定した後、

ac(j)=fac(ac(i),ac(h),…,ac(a))

なる関係を有するように定義付けられた論理式をノード情報の小さい順から実行し、

st(j)=st(k) & fst(ac(i),ac(h),…,ac(a))　ただし、k⟩j,i,h,…,a

なる関係を有するように定義付けられた論理式を、ノード情報の大きい順から実行し、故障検出条件の評価を行う演算手段を設けたことを特徴とする組合せ論理回路検査データ発生装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、組合せ組合せ論理回路、及び組合せ論理回路で構成された集積回路に適当なデータの組合せを入力して、その論理回路の構成が所望の通りになっているか否かを検査する検査装置に供給するデータの発生装置に関するものである。

【0002】

【従来の技術】組合せ論理回路で構成された回路及び集積回路においては、誤配線、未配線、素子の不具合、電源電圧変動による雑音余裕の変化などにより、様々な故障が生じる可能性がある。これらの論理回路の故障は以下のように分類されている。

【0003】単一縮退故障（stack at high/low）・・・論理回路内の論理素子の入出力端子の1箇所が“H”または“L”に固定されて故障が生じる。

多重故障・・・複数の単一故障が同時に発生する。

線間短絡故障・・・論理素子は全て正常であって誤配線、ブリッジ等によって生じる。論理回路のfunctionそのものが変化する場合もある。

間欠故障・・・雑音余裕が小さくなったときに生じる。再現性に乏しい。

【0004】上記のような故障があるが、多重故障、短絡故障の数は、回路規模に対して指数関数的に増大する。このため、充分小さい回路以外は、処理に非現実的

な数になってしまう。回路に多重故障が生じたとしても、その大部分が単一縮退故障と同一の検出方法で検出可能である。また、間欠故障は、主に集積回路内部のトランジスタレベル及び、集積回路が実装された基盤の状態等に起因する。以上の理由から論理回路の故障は、全て単一縮退故障であるとして仮定され、様々の故障検出方法が提案されている。

【０００５】組合せ論理回路では、１つの入力系列が決定すると一意に出力系列が決定する。そこで故障が生じていない回路と、故障が生じた回路の入出力関係を比較することにより故障検出を行うことが可能である。回路の入出力関係から回路の故障を検出するための条件は以下の通りである。

１）故障箇所の故障状態と反対の状態になるような入力パターンであること。

２）故障箇所の故障状態が回路の出力部まで伝播すること。

【０００６】図５は、組合せ論理回路におけるノードｄにおいて“Ｌ”縮退故障（stack atlow）が生じた場合を模式的に表す図である。上記故障を検出するためには、上記条件１）からｄが“Ｈ”であるための（Ａ，Ｂ）＝（Ｈ，Ｈ）である入力パターンが必要である。また条件２）からノードｅは“Ｌ”すなわちＣ＝“Ｈ”である必要がある。したがって、ノードｄのstack at lowを検出するためには、次のようなベクトルによるテストを試行する。

（Ａ，Ｂ，Ｃ，Ｙ）＝（Ｈ，Ｈ，Ｈ，Ｈ）

上記ベクトルによる試行が成功しないとき、回路内のノードｄにおいてstackat low故障が生じている可能性がある。

【０００７】これらの故障を検出するためには、入力の全組合せを試行し出力結果をモニターすればよいが、検査対象が多数あり、試行対象が入力端子を多数有するときには膨大な試行回数を必要とする。このため少ない試行回数で、検査対象の組合せ論理回路及び集積回路の検査を行えるのが望ましい。

【０００８】そこで、単一縮退故障のみに着目すると、上記テストにおいてノードｄのstack at lowの他、ノードｂ，ｃ，ｆのstack at high も検出可能である。したがって、故障箇所を限定する必要がなく、組合せ論理回路の良否のみを検出する場合は、入力の全組合せをテストする必要がない。

【０００９】組合せ論理回路の全ノードのstack at high/low を少なくとも１回検出し、テスト回数を最小化するためには、以下のような２つの方法がある。

【００１０】第１の方法は、あるノードについてstack at high 、stack at lowが検出されるテストを各々列ベクトルにする。これらの全故障ベクトル間で次の演算を行い、多項式を計算し積和の形にする。このうち最小の次数を有する積項が必要最小テストベクトルとなる。以

下具体例を挙げて説明する。

【００１１】図６は、図５の論理回路の入力全組合せを表す図である。ＡＢＣは論理回路の入力端子、ａｃ（）は各ノードの状態、ｓｔ（）はそのノードの検出可能な故障状態を表しており、“０”はそのノードの“Ｌ”縮退故障（stack at low）、“１”はそのノードの“Ｈ”縮退故障（stack at high ）、“・”はそのノード縮退故障が検出不可能なことを表す。

【００１２】この結果ｓｔ（）から、ノード及び、故障状態で故障検出が可能なテストベクトルを要素とする図７にある列ベクトルａｈ～ｆｌをつくる。ベクトルａｈはノードａにおける“Ｈ”縮退故障が検出可能なテストの集まりである。これらのベクトル内の各要素を和結合された項として、全ベクトル間で乗算する。但し、ここで行なわれる乗算は、同じ要素間で乗算が行なわれても、同じ項間で加算が行なわれても結果は変わらないとする。

【００１３】上記演算の結果、乗算で結合されているテストベクトルの組合せの各々が、全ノードの単一縮退故障を検出可能であり、この項の次数が最小の組合せが、最小ベクトル数で全ノードの単一縮退故障を検出可能な組合せである。

【００１４】Tmin = ah * bh * ch * dh * eh * fh * al * bl * cl * dl * el * fl

= t3 * t7 * t5 * t7 * (t0 + t2 + t4) * (t1 + t3 + t5) * (t1 + t3+ t5) ** t7 * (t0 + t2 + t4) * (t1 + t3 + t5) * (t0 + t2 + t4 + t6 + t7)* (t1 + t3 + t5)

【００１５】以下ｔ、積演算の印＊を省略する。

= 357 (0 + 2 + 4) (1 + 3 + 5) {(0 + 2 + 4) + (6 + 7) }

= 357 {(0 + 2 + 4) (1 + 3 + 5) + (0 + 2 + 4) (1 + 3 + 5) (6 + 7)}

【００１６】以下、積算項間の＋印を省略する。

= 357 {(01 03 05 12 23 25 14 34 45) + (01 03 05 12 23 25 14 34 45) (6 + 7) }

= 357 {(01 03 05 12 23 25 14 34 45) + (016 036 056 126 236 256 146 345 456 017 037 057 127 237 257 147 347 457)}

= 01357 0357 0357 12357 2357 2357 13457 3457 3457 013567 03567 03567 123567 23567 23567 134567 34567 34567 01357 0357 0357 12357 2357 2357 13457 13457 3457

【００１７】上記積項のうち同一なものを除き、

= 01357 0357 12357 2357 13457 3457 013567 03567 123567 23567 134567 34567 13457

上記積項でまとめられた試行の組合せの各々が、全ノードの単一縮退故障を少なくとも１回づつ検出する。このうち次数が最小なものは、

(t0, t3, t5, t7) (t2, t3, t5, t7) (t3, t4, t5, t7)

5

で、最小テストベクトル数は、4である。

【0018】組合せ論理回路の全ノードのstack at high/low を少なくとも1回検出し、テスト回数を減少するための第2の方法は、まず論理回路を構成している論理素子の出力ノードを番号付けする。各々の出力ノードは、回路の出力端子または、他の論理回路の入力端子と接続されており、これによってこの出力ノードに得られる出力が、論理素子によって決定される前段の出力ノードの関数として定義される。上記操作を入力ノードは入力端子として定義し、後段のノードが論理回路の出力端子に接続されたノードになるまで繰り返す。

【0019】ノードjがノードk，l，m，n，…，zによって以下のように定義されているとき番号付けは以下の条件が守られるように行なわれる。

ac(j)=f(ac(k),ac(l),ac(m),ac(n),…,ac(z))

ac(j) はノードjの状態を表す。このとき、j(k,l,m,n,…,z

【0020】上記条件より、入力として定義されたノードに入力ベクトルを与え、ノード番号順に論理演算fを実行すると、入力ベクトルによって一意に決定される論理回路の各ノードの状態ベクトルが得られる。これによって前記故障検出条件の1）、1つの入力ベクトルに対して、各ノードにおける検出可能な故障状態が明らかになる。

【0021】更に故障検出条件の2）、故障状態が出力まで伝播するための条件を満たす必要がある。この条件は以下のように表される。ノードj及び、i，h，g，…，aがある論理素子の入力、ノードkがその出力となっているとき、

st(j)=st(k) & fac(ac(i),ac(h),ac(g),…,ac(a))

st(j) はノードjの状態が出力まで伝播可能であるか否かを表す。

ただし、k)j,i,h,g,…,a

【0022】論理素子の出力ノードのうち、回路の出力と接続されているノードはそのノード自身を常に観測可能であるため、st(k) は常に真（“H”とする）となる。またノードkが出力となっている論理素子への入力ノードjの状態が出力kへ伝播するためには、j以外の前記論理素子への入力i，h，g，…，aが以下の条件を満たす必要がある。

【0023】i，h，g，…，a＝0　（前記論理素子がOR，NORのとき）

i，h，g，…，a＝1　（前記論理素子がAND，NANDのとき）

i，h＝X　（前記論理素子がX－OR，X－NORのとき）

i＝X　（前記論理素子がINVのとき）　（Xは任意の値を表す。）

【0024】上記条件を表したものがfac() である。上記演算をノード番号の逆順で行うことによりある入力ベクトルに対して故障伝播可能なノードが定まる。上記操作により、ある入力ベクトルに対して故障検出可能なノードとその故障状態が定まる。これを①まず入力全組合せに対して実行し、故障検出数が最大となる入力ベクトルを抽出する。②更に前記の入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる入力ベクトルを前記入力ベクトルを除いた他のベクトルの中から抽出する。③前記操作により抽出された入力ベクトル群により検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる入力ベクトルを抽出する。上記操作③を、新たに故障検出可能なノードが現れなくなるまで繰り返す。これによって組合せ論理回路に冗長な部分がないとき、抽出されたベクトル群で全ノードの単一縮退故障が検出可能である。

6

【0025】

【発明が解決しようとする課題】しかしながら、上記従来例の第1の方法では、以下のような欠点があった。一例として回路内総ノード数m、入力端子数nの組合せ論理回路において上記手法で試行回数が最小になるテストベクトルを求めようとすると、あるノードのstack at high または、stack at low検出可能なテストを表す列ベクトルの要素数は、1～2のn乗、列ベクトル数は2mになる。したがって、前述の積和項演算は、1～2のn乗の要素数を持つ和項2m個の乗算となり、最小テストベクトルを得るのに非常に大規模な論理積和演算を必要とするという欠点があった。

【0026】また、上記第2の方法では、論理回路の構成を表す各ノード間の論理関数の他に故障箇所が出力まで伝播する条件をノード毎に定義する必要があった。

【0027】更に、上記第2の方法では、各々の出力ノードは、回路の出力端子または、他の論理回路の入力端子と接続されており、これによってこの出力ノードに得られる出力が、論理素子によって決定される前段の出力ノードの関数として定義される。上記操作を入力ノードは入力端子として定義し、後段のノードが論理回路の出力端子に接続されたノードになるまで繰り返す。ノードjがノードk，l，mn，…，zによって以下のように定義されているとき番号付けを以下の条件が守られるように行う必要があった。

ac(j)=f(ac(k),ac(l),ac(m),ac(n),…,ac(z))

ac(j) はノードjの状態を表す。このとき、j(k,l,m,n,…,z

【0028】本発明は、上記のような問題を解決するためになされたもので、演算処理の簡単なデータ発生装置を得ることを目的としている。

【0029】

【課題を解決するための手段】第1の発明は、複数の論理素子により構成された組合せ論理回路に対する全入力の組合せを試行し、故障検出数が最大となる第1の入力ベクトルを抽出し、更に上記第1の入力ベクトルにより

7

検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる第２の入力ベクトルを抽出する試行を行い、第ｎ－１，ｎ－２，…，２，１の入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる第ｎの入力ベクトルを抽出することにより得られたデータを、上記組合せ論理回路の回路構成が所望の通りになっているか否かを検査する検査装置に供給する組合せ論理回路検査データ発生装置において、上記回路構成の定義を上記論理素子各々の入出力関係について行い、その際の順序づけを、各論理素子の入出力点をａ，…，ｈ，ｉとしたとき、

ac(j)=fac(ac(i),ac(h),…,ac(a))　ただし、j⟩a,…,h,i

なる関係を有するように定義付けられた論理式を用いてデータの発生を行う演算手段を設けたことを特徴とするものである。

【００３０】第２の発明は、複数の論理素子により構成された組合せ論理回路に対する全入力の組合せを試行し、故障検出数が最大となる第１の入力ベクトルを抽出し、更に上記第１の入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる第２の入力ベクトルを抽出する試行を行い、第ｎ－１，ｎ－２，…，２，１の入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる第ｎの入力ベクトルを抽出することにより得られたデータを、上記組合せ論理回路の回路構成が所望の通りになっているか否かを検査する検査装置に供給する組合せ論理回路検査データ発生装置において、上記回路構成の定義を上記論理素子各々の入出力関係について行い、その際の順序づけを、各論理素子の入出力点をａ，…，ｈ，ｉとしたとき、

ac(j)=fac(ac(i),ac(h),…,ac(a))　ただし、j⟩a,…,h,i

なる関係を有するように定義付けられた論理式を用いてデータの発生を行い、

st(j)=st(k) & fst(ac(i),ac(h),…,ac(a))　ただし、k⟩j,i,h,…,a

なる関係を有するように定義付けられた論理式を用いて条件の評価を行う演算手段を設けたことを特徴とするものである。

【００３１】第３の発明は、複数の論理素子により構成された組合せ論理回路に対する全入力の組合せを試行し、故障検出数が最大となる第１の入力ベクトルを抽出し、更に上記第１の入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる第２の入力ベクトルを抽出する試行を行い、第ｎ－１，ｎ－２，…，２，１の入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる第ｎの入力ベクトルを抽出することにより得られたデータを、上記組合せ論理回路の回路構成が所望の

8

通りになっているか否かを検査する検査装置に供給する組合せ論理回路検査データ発生装置において、上記回路構成の定義を上記論理素子各々の入出力関係について行い、各論理素子の入出力点としてのノードａ，ｂ，…，ｈ，ｉにノード情報の記憶領域を有し、入力端子に接続されたノードを初期化したのち、定義された論理演算の実行順として、上記ノードに論理素子を介して接続されているノード群のノード情報が全て確定しているとき、上記ノード群のノード情報の中で最大のノード情報を単調増加させた値をそのノードのノード情報として記憶し、上記ノード情報が全ノードで確定した後、

ac(j)=fac(ac(i),ac(h),…,ac(a))

なる関係を有するように定義付けられた論理式をノード情報の小さい順から実行し、

st(j)=st(k) & fst(ac(i),ac(h),…,ac(a))　ただし、k⟩j,i,h,…,a

なる関係を有するように定義付けられた論理式を、ノード情報の大きい順から実行し、故障検出条件の評価を行う演算手段を設けたことを特徴とするものである。

【００３２】

【作用】第１の発明によれば、最大場合の数が入力の全組合せである試行を行うことにより、回路内の全ノードの単一縮退故障を少なくとも１回検出するパターンを生成することができる。

【００３３】第２の発明によれば、最大場合の数が入力の全組合せである試行を行うことにより、回路内の全ノードの単一縮退故障を少なくとも１回検出するパターンを生成する際に、回路の故障検出条件を自動発生する。

【００３４】第３の発明によれば、回路内の全ノードの単一縮退故障を少なくとも１回検出するパターンを生成する際に、回路の定義を論理素子の入出力関係をランダムに記述できるようにし、回路定義を容易ならしめる。

【００３５】

【実施例】以下、第１～３の発明の各実施例を図について説明する。

【００３６】第１～３の発明においては、論理回路の構成は以下のようにして定義される。まず論理回路を構成している論理素子の出力ノードを番号付けする。各々の出力ノードは、回路の出力端子または、他の論理回路の入力端子と接続されており、これによってこの出力ノードに得られる出力が、前段の出力ノードの関数として定義される。上記操作を入力ノードは入力端子として定義し、後段のノードが論理回路の出力端子に接続されたノードになるまで繰り返す。

【００３７】ノードｊがノードｋ，ｌ，ｍ，ｎ，…，ｚによって以下のように定義されているとき番号付けは以下の条件ａ）が守られるように行なわれる。

ac(j)=f(ac(k),ac(l),ac(m),ac(n),…,ac(z))

ac(j)はノードｊの状態を表す。このとき、j⟨k,l,m,n,…,z

9

【００３８】上記条件より、入力として定義されたノードに入力ベクトルを与え、ノード番号順に論理演算ｆを実行すると、入力ベクトルによって一意に決定される論理回路の各ノードの状態ベクトルが得られる。これによって前記した従来例に示された故障検出条件の１）、１つの入力ベクトルに対して、各ノードにおける検出可能な故障状態が明らかになる。

【００３９】更に故障検出条件の２）、故障状態が出力まで伝播するための条件を満たす必要がある。この条件は以下のように表される。ノードｊ及び、ｉ，ｈ，ｇ，…，ａがある論理素子の入力、ノードｋがその出力となっているとき、

st(j)=st(k) & fac(ac(i),ac(h),ac(g),…,ac(a))

st(j) はノードｊの状態が出力まで伝播可能であるか否かを表す。

ただし、k)j　　（条件ｂ））

【００４０】論理素子の出力ノードのうち、回路の出力と接続されているノードはそのノード自身を常に観測可能であるため、st(k) は常に真（“H”とする）となる。このノードｋが出力となっている論理素子への入力ノードｊの状態が出力ｋへ伝播するためには、ｊ以外の前記論理素子への入力ｉ，ｈ，ｇ，…，ａが以下の条件ｃ）を満たす必要がある。

【００４１】ｉ，ｈ，ｇ，…，ａ＝０　（前記論理素子がＯＲ，ＮＯＲのとき）

ｉ，ｈ，ｇ，…，ａ＝１　（前記論理素子がＡＮＤ，ＮＡＮＤのとき）

ｉ，ｈ＝Ｘ　（前記論理素子がＸ－ＯＲ，Ｘ－ＮＯＲのとき）

ｉ＝Ｘ　（前記論理素子がＩＮＶのとき）　（Ｘは任意の値を表す。）

【００４２】上記条件を表したものがfac である。上記演算をノード番号の逆順で行うことによりある入力ベクトルに対して故障伝播可能なノードが定まる。上記操作により、ある入力ベクトルに対して故障検出可能なノードとその故障状態が定まる。これを①まず入力全組合せに対して実行し、故障検出数が最大となる入力ベクトルを求める。②更に前記入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる入力ベクトルを前記入力ベクトルを除いた他のベクトルの中から求める。上記操作②を、新たに故障検出可能なノードが現れなくなるまで繰り返す。これによって組合せ論理回路に冗長な部分がないときは、全ノードの単一縮退故障が検出可能である。

【００４３】以下、実施例を用いて第１の発明を詳述する。

【００４４】図３は、図５の各ノード間の論理入出力関係を定義したものである。ノードａ，ｂ，ｃ，ｄ，ｅ，ｆはそれぞれac(0),ac(1),ac(2),ac(3),ac(4),ac(5) に対応しており、b(n)は、入力端子に直結したノードに入

10

力される０～２のべき乗のそれぞれの桁を入力し、ノード番号の小さい順に下記に定義された論理演算を実行する。これによってある入力ベクトルが得られたとき全ノードの状態が定まり、前記故障検出を行うための条件１）、各ノードにおいてどの故障状態が検出可能かが求められる。

【００４５】図４のst()は、図５の各ノードにおける、前記故障検出条件２）、故障箇所の状態が出力まで伝播するための条件を表す図である。論理回路の出力端子に直結されたノードは常時モニター可能なので、このノードの条件２）は常に“１”になる。この式のst() and以下の部分が前式のfac() に相当する。他のノードの条件は着目ノードと同一の論理素子に入力されている他の入力ノード状態と出力ノードによって、図４のように定義される。この条件式をノード番号が大きい順に評価していくことで、ある入力ベクトルが与えられたとき、故障検出可能なノードが明らかになる。

【００４６】したがって、ノードｊにおいて、

st(j)=1 and ac(j)=0 なら、stack at high 検出

st(j)=1 and ac(j)=1 なら、stack at low　検出

st(j)=0 and ac(j)=x なら、故障検出不可能

となる。

【００４７】上記処理を全入力ベクトルについて行ったのが図６であり、“０”はそのノードの“Ｌ”縮退故障（stack at low）、“１”はそのノードの“Ｈ”縮退故障（stack at high ）、“・”はそのノードの縮退故障が検出されないことを表す。これらのテスト結果の中で故障検出数が最も大きいのはテスト３であり、更に前記入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる入力ベクトルを前記入力ベクトルを除いた他のベクトルの中から求め、テスト７が得られる。さらに故障検出可能なノードが現れなくなるまでテストベクトルの抽出を繰り返し、テスト０、５が得られる。上記組合せ論理回路には冗長な部分がないので、前記操作で抽出されたテストで全ノードの単一縮退故障が検出可能である。

【００４８】図１は、上述した原理及び後述する第２、第３の発明によるデータ発生装置を概略的に示すブロック図であり、図１において、１は第１、第２の入力ベクトルを抽出する入力ベクトル抽出部、２は入力ベクトルの抽出結果に基づき予め定義された論理式を用いてデータを発生する演算部、３はデータを供給される検査装置、４は検査対象の組合せ論理回路である。

【００４９】次に、図２のフローチャートと共に概略的な動作を説明する。まず、ステップＳ１で演算部２により回路構成を定義し、さらに前述したac(j)=f(ac(i),ac(h),…,ac(a)) の関係となるような論理式の定義を行う。次にステップＳ２で入力ベクトル抽出部１において、検出可能なノードの故障状態が最大になるような第１のテスト（入力）ベクトルを選出し、ステップＳ３で

11

上記テストベクトルにより検出されるノードの故障状態を除いた上で、更に検出可能なノードの故障状態数が最大となる第２のテストベクトルを抽出し、全ノードの全故障状態が検出されるまで、第１から第ｎ－１までのテストベクトルにより検出されるノードの故障状態を除いた上で、ステップＳ４により更に検出可能なノードの故障状態数が最大となる第ｎのテストベクトルを抽出されるまでステップＳ３の処理を繰り返して、ステップＳ５で全ノードの故障状態を検出するようなテストデータを発生させる。

【００５０】図８～図１１は、処理を詳細を示すフローチャートである。

【００５１】次に、実施例を用いて第２の発明を詳述する。

【００５２】図１２は、図５の各ノード間の論理入出力関係を定義したものであり、図３は図１２で表された論理回路の論理素子毎の入出力関係を表している。上記の形式で表された論理関数は、予め本発明処理フロー内に関数として確保されている上記論理演算のどれに該当するか評価され、該当する論理演算の例えば先頭アドレスなどの処理フローを決定する情報を、関数実行順として配列fun[] に確保し、この論理素子のある入力ノードが出力に伝播するための条件を評価する関数を、前述の条件ｃ）の中から抽出し、例えば評価関数処理ルーチン先頭アドレスなど、処理フロー先頭アドレスを関数実行順として、配列funst[] に確保する。

【００５３】更に入出力関係を確認するために、ある出力ノードｊに接続された論理素子に入力されるノードをその出力ノードに関連付けて配列intm[]に確保し、各ノードの状態を求めるほか、ある入力ノードｊが論理素子を介して接続された出力ノードを入力ノードに関連して、配列ottm[]に確保しておき、故障状態の伝播の評価に使用する。

【００５４】ノード状態の決定は以下のように行う。前述の条件ａ）を満たすように論理関数を実行すると、第１に入力端子に直結されたノードにある入力パターンが代入され、これと接続されているノードの状態が定まる。したがって、条件ａ）を満たす順番で実行される論理演算は、常に入力ノードが定まった状態で呼び出される。よって、決定したいノードがｊの時、ｊが得られる論理関数fun(j)、その入力ノードintm[j] を呼び出し、ｊの状態が決定される。

【００５５】ノード状態の伝播条件の決定は、以下のように行なわれる。組合せ論理回路にある入力パターンが与えられ、回路内の全ノード状態が決定されたのち、条件ｂ）を大きいノード番号順に評価する。これによって第１に出力端子に直結されたノードの出力伝播条件が評価される。出力端子は常にモニター可能なので、このノードの出力伝播条件st()は“真”であり、このときのノード状態の反対の状態の縮退故障の検出が可能である。

12

【００５６】評価するノードが出力端子に直結されていない場合には、ottm[j] を参照してノードｊが入力となっている論理素子の出力ノードが出力端子まで伝播可能となっているかst(ottm[j]) を評価する。st(ottm[j]) は条件ｂ）に従い出力に近い階層から評価されており、常に出力ノードが伝播可能であるか評価し終わった段階で呼び出される。

【００５７】さらにintm[ottm[j]] を引数としてfunst(j)を実行し、ノードｊの状態がノードｊが入力端子となっている論理素子を伝播可能かどうか評価する。上記st(ottm[j]) とfunst(j)の論理積が真であるときノードｊの故障状態が伝播可能であり、このときのノードｊの反対の状態の縮退故障が検出可能になる。

【００５８】また、図３、図６について前述したように、入力ベクトルが得られたとき全ノードの状態が定まり、前記故障検出を行うための条件１）、各ノードにおいてどの故障状態が検出可能かが求められる。さらに、図４、図５について前述したように、上記組合せ論理回路には冗長な部分がないので、前記操作で抽出されたテストで全ノードの単一縮退故障検出可能である。

【００５９】図１３～図１６は処理の詳細を示すフローチャートである。

【００６０】次に、実施例を用いて第３の発明を詳述する。

【００６１】前述した図１２、図５、図３等に関する上記各論理演算の階層化は以下のようにして行う。

①全ノードの中から入力端子として定義されているノードｊを抽出して階層“０”を各ノードの階層を記憶する配列lay[j]に記憶する。②次に、あるノードｊに論理素子を介在して入力を供給しているノードの階層化情報lay[tm[j]]を参照して、これらの入力ノードの階層付けが全て終了しているか否かを判別する。全ての入力ノードの階層付けが終了している場合には、それらの中で最も大きいノードに“１”を加えた数をそのノードの階層とし、階層化終了情報をlayf[j]に入力する。全ての入力ノードの階層付けが終了していない場合には、何もしないで次のノードに対して同様の操作を行う。上記②の操作を全ノードの階層化が行なわれるまで繰り返す。以上の処理によりノードの階層化が終了する。これによって論理演算の実行順を階層番号lay[j]の小さい順で行うことにより回路が定義された順に関わらず、ある入力パターンに対して一意に決定される各ノード状態、及び出力が得られる。

【００６２】図１８は、処理の詳細を示すフローチャートであり、ステップＳ１８１～Ｓ１８６のみが図１３と異なっており、ステップＳ１３１～Ｓ１３４は、図１３と同じである。また、ステップＳ１３５～Ｓ１６０は、図１４～図１６と同じであるので省略した。

【００６３】尚、第１～第３の発明における上記処理を、予め与えられた入力ベクトルについて行い、全入力

ベクトルの試行が終了した段階で、処理を終了することにより、与えられた入力ベクトルにより回路内の故障状態がどのくらい検出可能かを知ることも可能である。

【0064】また、回路及び集積回路内に順序動作を行う素子がある場合でも、回路の内部状態が入力ベクトルによって一意に定まる構成であれば出力ベクトルと入力ベクトルの得られる時間差を除去したテストベクトルで上記処理を行うことで、内部の組合せ論理素子の検査が可能である。この場合、順序動作素子がフリップフロップなどで構成され、論理素子外部から供給されるクロックにより、組合せ論理素子のデータを転送する手段として使用されている場合は上記処理により発生されたベクトルで試行することにより動作を診断しても、実用上差し支えない。

【0065】

【発明の効果】以上説明したように、第1の発明によれば、複数の論理素子により構成された組合せ論理回路において、①入力全組合せをシミュレーションし、故障検出数が最大となる入力ベクトルを求める。②更に前記入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる入力ベクトルを前記入力ベクトルを除いた他のベクトルの中から求める試行を行うことにより、最大場合の数が入力の全組合せである試行を行うことにより、回路内の全ノードの単一縮退故障を少なくとも1回検出するパターンを生成することができる。

【0066】また、第2の発明によれば、組合せ論理回路において、①入力全組合せをシミュレーションし、故障検出数が最大となる入力ベクトルを求める。②更に前記入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる入力ベクトルを前記入力ベクトルを除いた他のベクトルの中から求める。③前記までの操作により抽出された入力ベクトル群により検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる入力ベクトルを抽出する。③の動作を新たに検出可能な故障状態が発生しなくなるまで繰り返すことにより、最大場合の数が入力の全組合せである試行を行うことにより、回路内の全ノードの単一縮退故障を少なくとも1回検出するパターンを生成する際に、回路の故障検出条件を自動発生するという利点がある。

【0067】第3の発明によれば、組合せ論理回路において、①入力全組合せをシミュレーションし、故障検出数が最大となる入力ベクトルを求める。②更に前記入力ベクトルにより検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる入力ベクトルを前記入力ベクトルを除いた他のベクトルの中から求める。③前記までの操作により抽出された入力ベクトル群により検出可能な故障状態を除いた故障状態空間の中で故障検出数が最大となる入力ベクトルを抽出する。③の動作を新たに検出可能な故障状態が発生しなくなるまで繰り返すことにより、最大場合の数が入力の全組合せである試行を行うことにより、回路内の全ノードの単一縮退故障を少なくとも1回検出するパターンを生成する際に、回路の定義を論理素子の入出力関係をランダムに記述できるようにし、回路定義を容易ならしめる効果がある。

【図面の簡単な説明】

【図1】本発明の実施例によるデータ発生装置を概略的に示すブロック図である。

【図2】動作を概略的に示すフローチャートである。

【図3】組合せ論理回路の各ノードの関係を示す説明図である。

【図4】組合せ論理回路の各ノードの故障検出条件を示す説明図である。

【図5】本発明及び従来例の処理方法を説明するための組合せ論理回路の一例を示す構成図である。

【図6】組合せ論理回路の入力全組合せを示す説明図である。

【図7】組合せ論理回路の各ノードの故障検出可能なテストベクトルを要素とする列ベクトルを示す説明図である。

【図8】第1の発明の実施例による処理を示すフローチャートである。

【図9】上記フローチャートの続きを示すフローチャートである。

【図10】上記フローチャートの続きを示すフローチャートである。

【図11】上記フローチャートの続きを示すフローチャートである。

【図12】組合せ論理回路の各ノードの入出力関係の定義の一例を示す説明図である。

【図13】第2の発明の実施例による処理を示すフローチャートである。

【図14】上記フローチャートの続きを示すフローチャートである。

【図15】上記フローチャートの続きを示すフローチャートである。

【図16】上記フローチャートの続きを示すフローチャートである。

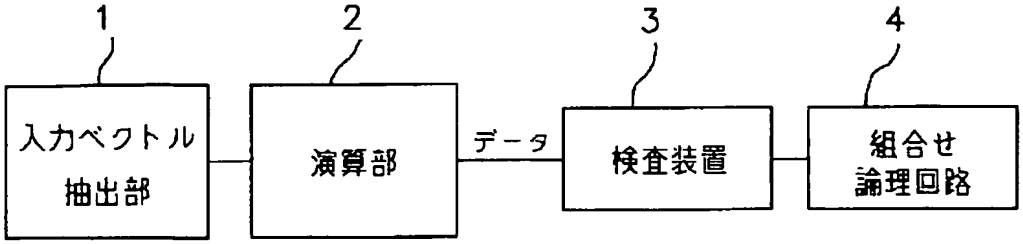
【図17】図13～図16の処理フロー内で判定、割当、評価される配列の内容を示す説明図である。

【図18】第3の発明の実施例による処理を示すフローチャートである。

【符号の説明】

1　入力ベクトル抽出部
2　演算部
3　検査装置
4　組合せ論理回路

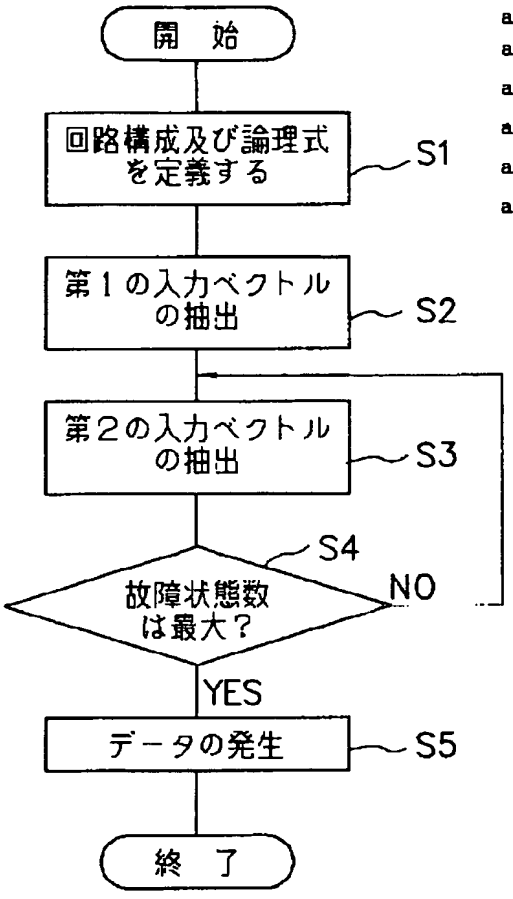
【図1】



【図12】

```
iput 3
inet 6
isup 1
ac000 inl 2
ac001 inl 1
ac002 inl 0
ac003 an2 0 1
ac004 ivl 2
ac005 or2 3 4
xxxx otl 5
```

【図2】



【図3】

```
ac(0) = b(2)
ac(1) = b(1)
ac(2) = b(0)
ac(3) = ac(0) and ac(1)
ac(4) = not ac(2)
ac(5) = ac(3) or ac(4)
```

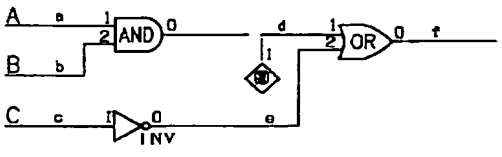
【図4】

```
st(5) = 1
st(4) = st(5) and (not ac(3))
st(3) = st(5) and (not ac(4))
st(2) = st4
st(1) = st(3) and ac(0)
st(0) = st(3) and ac(1)
```

【図6】

			ac()	st()
test	ABC	Y	abcdef	abcdef
t0	000	0	000011	··1·00
t1	001	0	001000	··0111
t2	010	0	010011	··1·00
t3	011	0	011000	1·0111
t4	100	0	100011	··1·00
t5	101	0	101000	·10111
t6	110	0	110111	·····0
t7	111	0	111101	00·0·0

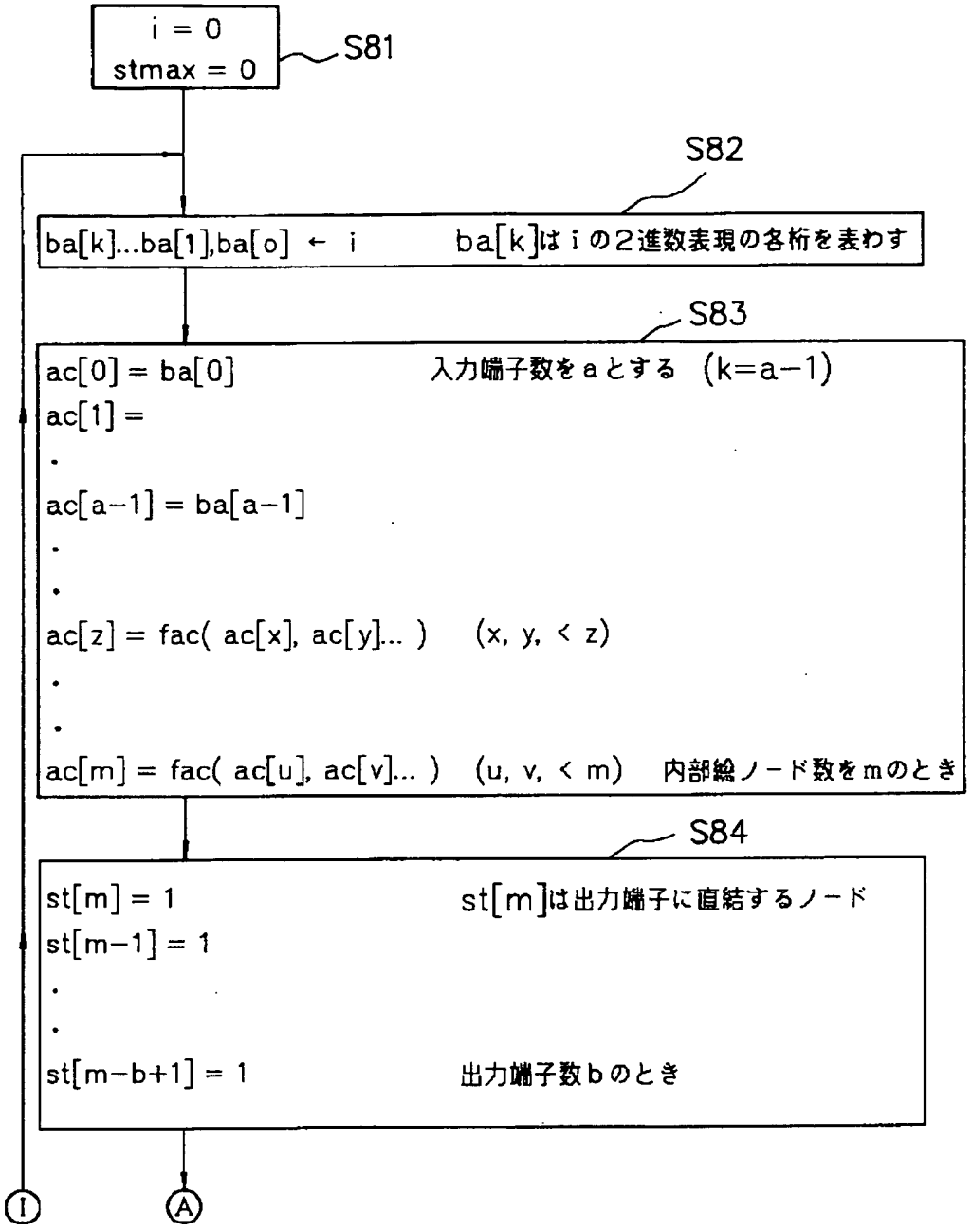
【図5】



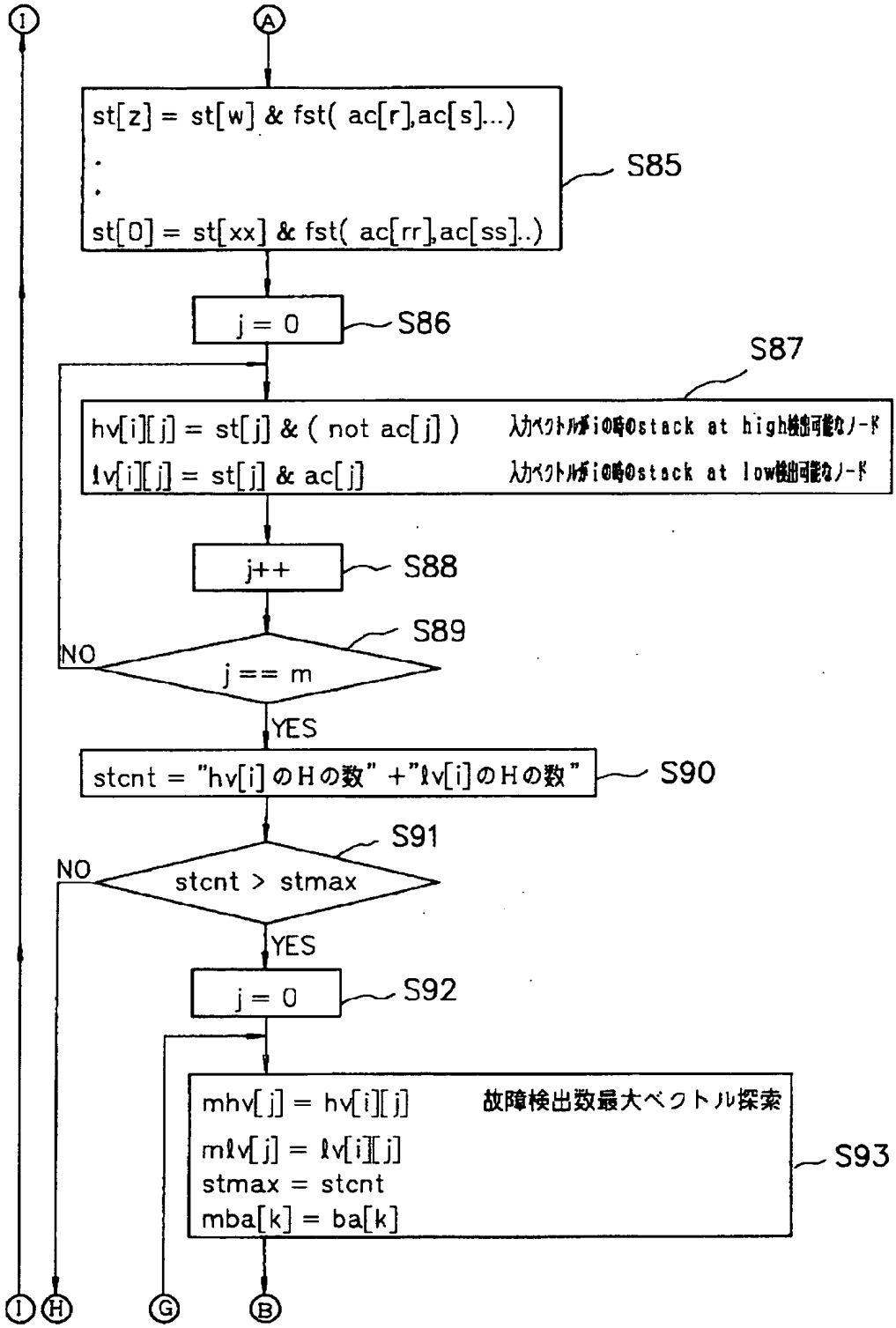
【図7】

```
ah = (        t3)
al = (                t7)
bh = (           t5)
bl = (                t7)
ch = (t0,  t2,  t4,)
cl = (  t1,  t3,  t5)
dh = (  t1,  t3,  t5)
Ydl = (               t7)
eh = (t0,  t2   t4,)
el = (  t1,  t3,  t5)
fh = (t0,  t2,  t4,  t6,t7)
fl = (  t1,  t3,  t5)
```

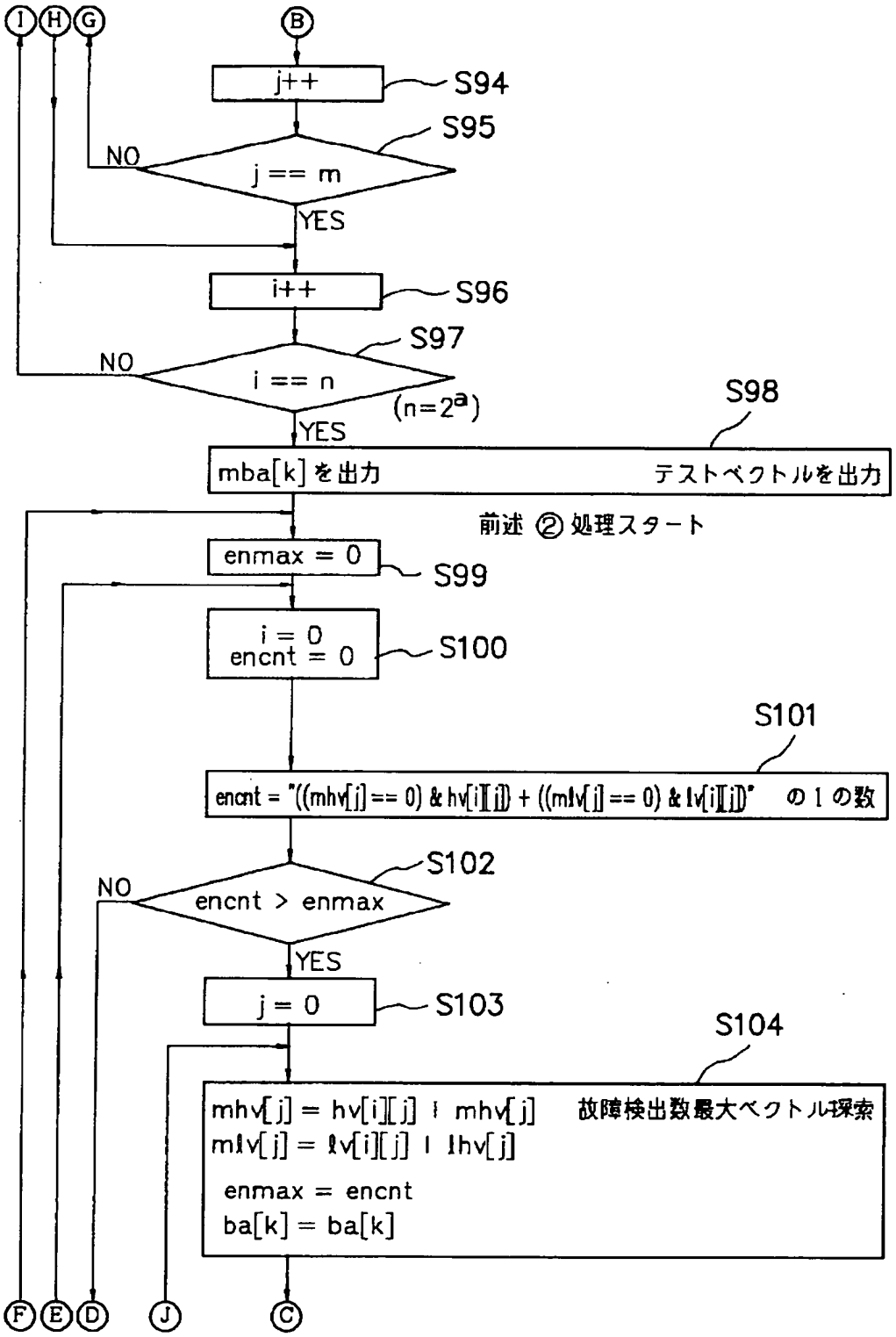
【図8】


i = 0
stmax = 0
S81
S82
ba[k]...ba[1],ba[o] ← i
ba[k]はiの2進数表現の各桁を表わす
S83
ac[0] = ba[0]
入力端子数をaとする (k=a−1)
ac[1] =
ac[a−1] = ba[a−1]
ac[z] = fac(ac[x], ac[y]...) (x, y, < z)
ac[m] = fac(ac[u], ac[v]...) (u, v, < m)
内部総ノード数をmのとき
S84
st[m] = 1
st[m]は出力端子に直結するノード
st[m−1] = 1
st[m−b+1] = 1
出力端子数bのとき
I
A

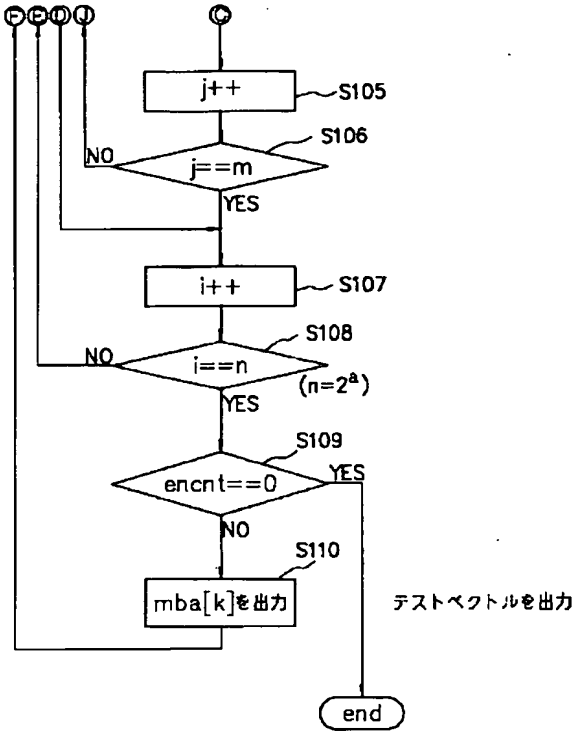
【図9】


I
A
st[z] = st[w] & fst(ac[r],ac[s]...)
st[0] = st[xx] & fst(ac[rr],ac[ss]..)
S85
j = 0
S86
S87
hv[i][j] = st[j] & (not ac[j])
lv[i][j] = st[j] & ac[j]
j++
S88
S89
NO
j == m
YES
stcnt = "hv[i]のHの数" +"lv[i]のHの数"
S90
S91
NO
stcnt > stmax
YES
j = 0
S92
mhv[j] = hv[i][j]
故障検出数最大ベクトル探索
mlv[j] = lv[i][j]
stmax = stcnt
mba[k] = ba[k]
S93
I
H
G
B

【図10】


I
H
G
B
j++
S94
S95
NO
j == m
YES
i++
S96
S97
NO
i == n
(n=2^a)
YES
S98
mba[k]を出力
テストベクトルを出力
前述 ② 処理スタート
enmax = 0
S99
i = 0
encnt = 0
S100
S101
encnt = "((mhv[j] == 0) & hv[i][j]) + ((mlv[j] == 0) & lv[i][j])" の1の数
S102
NO
encnt > enmax
YES
j = 0
S103
S104
mhv[j] = hv[i][j] | mhv[j]
mlv[j] = lv[i][j] | lhv[j]
故障検出数最大ベクトル探索
enmax = encnt
ba[k] = ba[k]
F
E
D
J
C

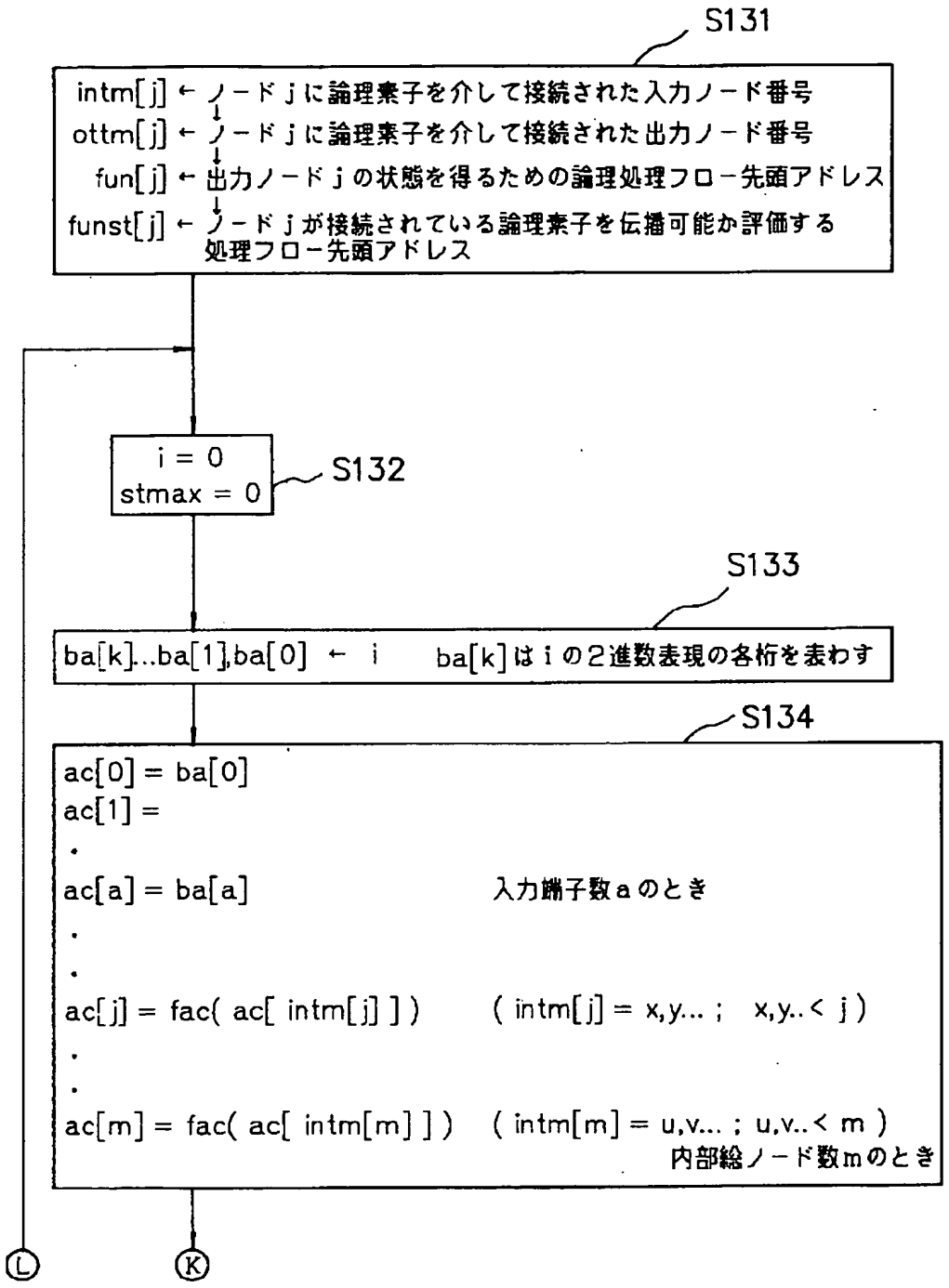
【図１１】



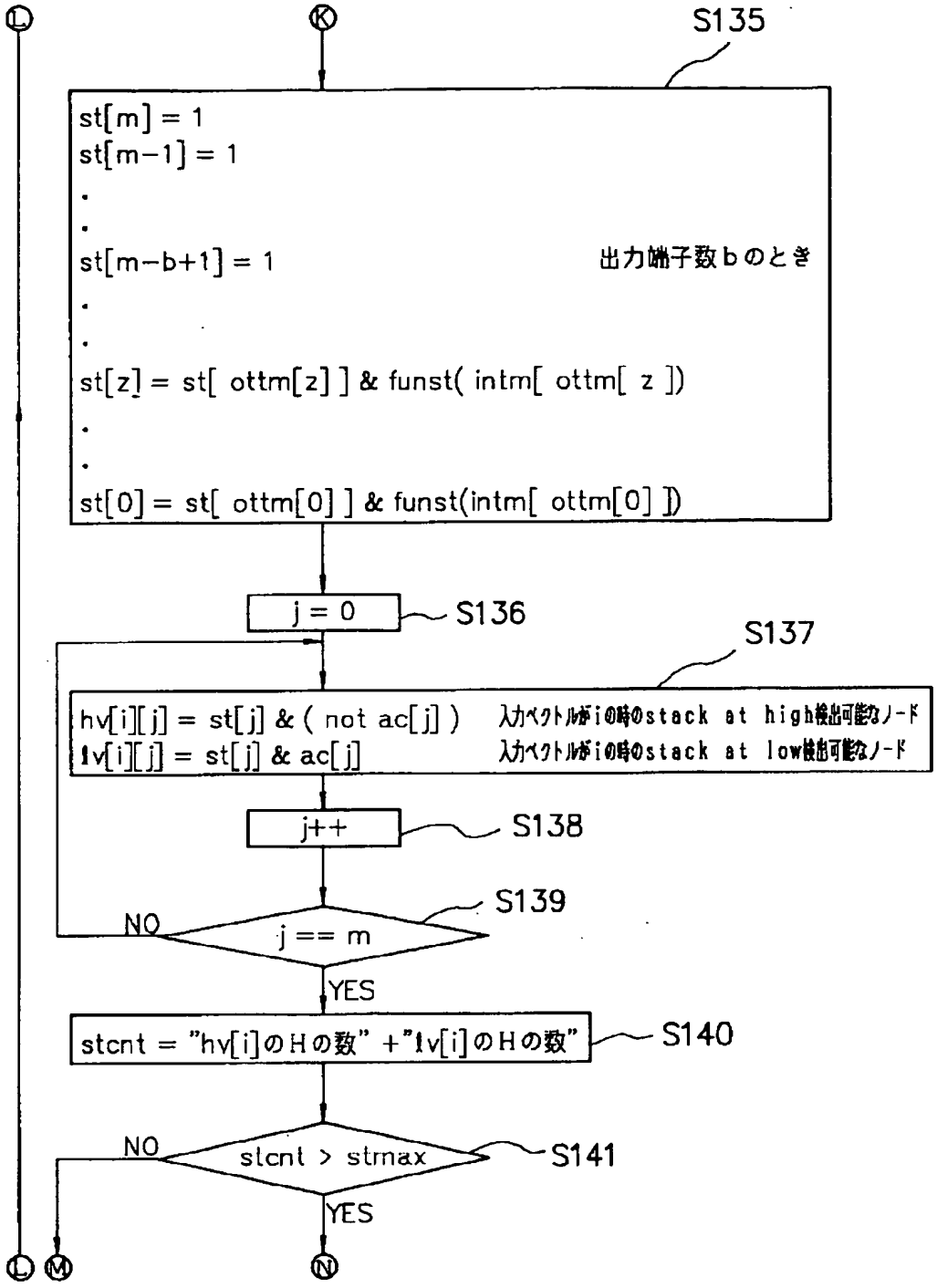
【図１７】

```
intm[0] = ba[2]
intm[1] = ba[1]
intm[2] = ba[0]
intm[3] = 0, 1
intm[4] = 2
intm[5] = 3, 4

ottm[0] = 3
ottm[1] = 3
ottm[2] = 4
ottm[3] = 5
ottm[5] = 5

m = abc ( binary ) のとき
fun[0] = a
fun[1] = b
fun[2] = c
fun[3] = and
Fun[4] = Inv
fun[5] = or

funst[5] = 1
funst[4] = stor      stor( i, j, k, . . . )  { = 1 ( i=j=k=0)
funst[3] = stor                                = 0 (!(i=j=k=0))
funst[2] = stand     stand( i, j, k, . . )   { = 1 ( i=j=k=1)
funst[1] = stand                               = 0 (!(i=j=k=1))
funst[0] = stinv     stinv( i ) = 1
```

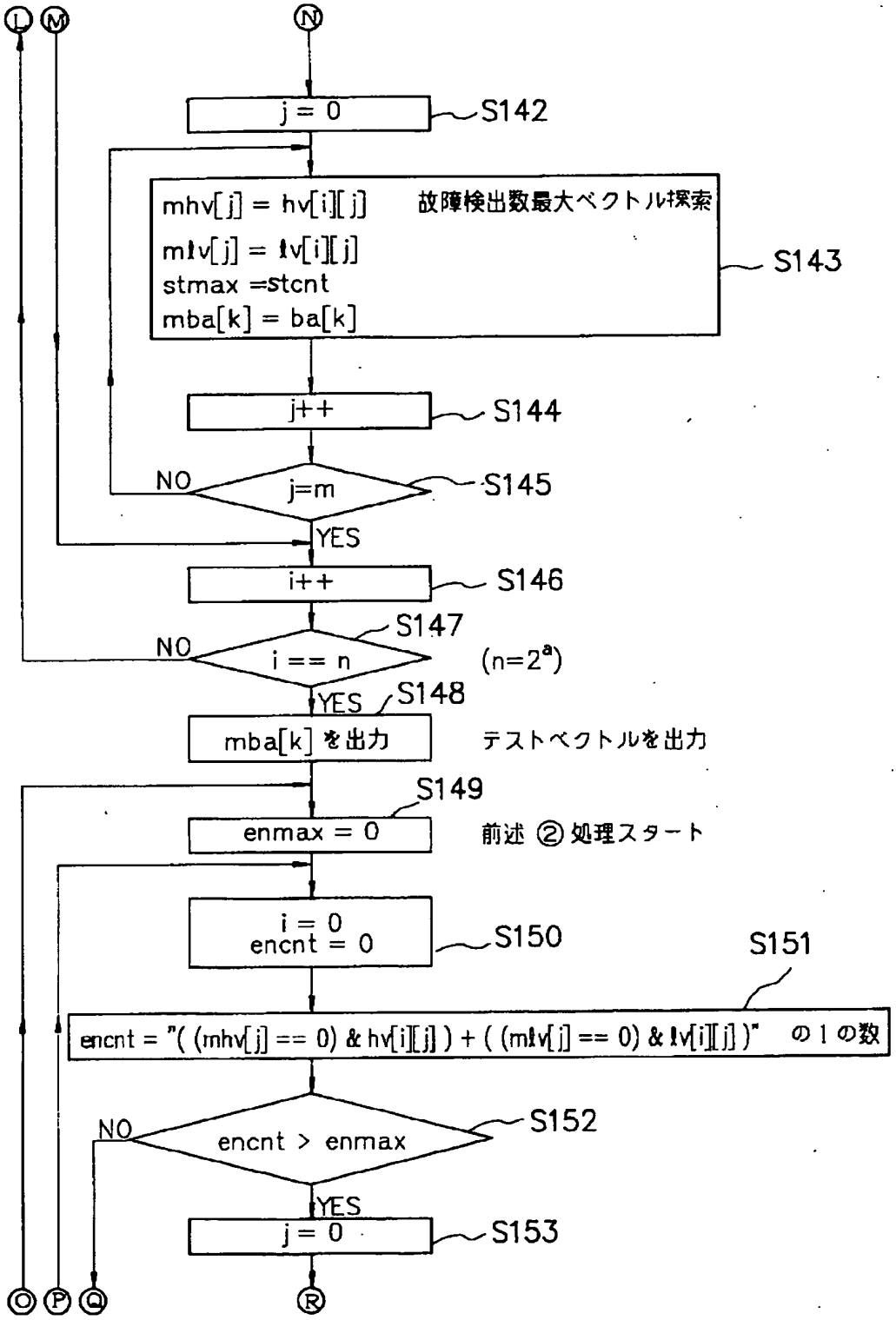
【図１３】


S131
intm[j] ← ノードｊに論理素子を介して接続された入力ノード番号
ottm[j] ← ノードｊに論理素子を介して接続された出力ノード番号
fun[j] ← 出力ノードｊの状態を得るための論理処理フロー先頭アドレス
funst[j] ← ノードｊが接続されている論理素子を伝播可能か評価する処理フロー先頭アドレス
i = 0
stmax = 0
S132
S133
ba[k]...ba[1],ba[0] ← i ba[k]はｉの2進数表現の各桁を表わす
S134
ac[0] = ba[0]
ac[1] =
ac[a] = ba[a] 入力端子数ａのとき
ac[j] = fac(ac[intm[j]]) (intm[j] = x,y... ; x,y..< j)
ac[m] = fac(ac[intm[m]]) (intm[m] = u,v... ; u,v..< m)
内部総ノード数ｍのとき
L
K

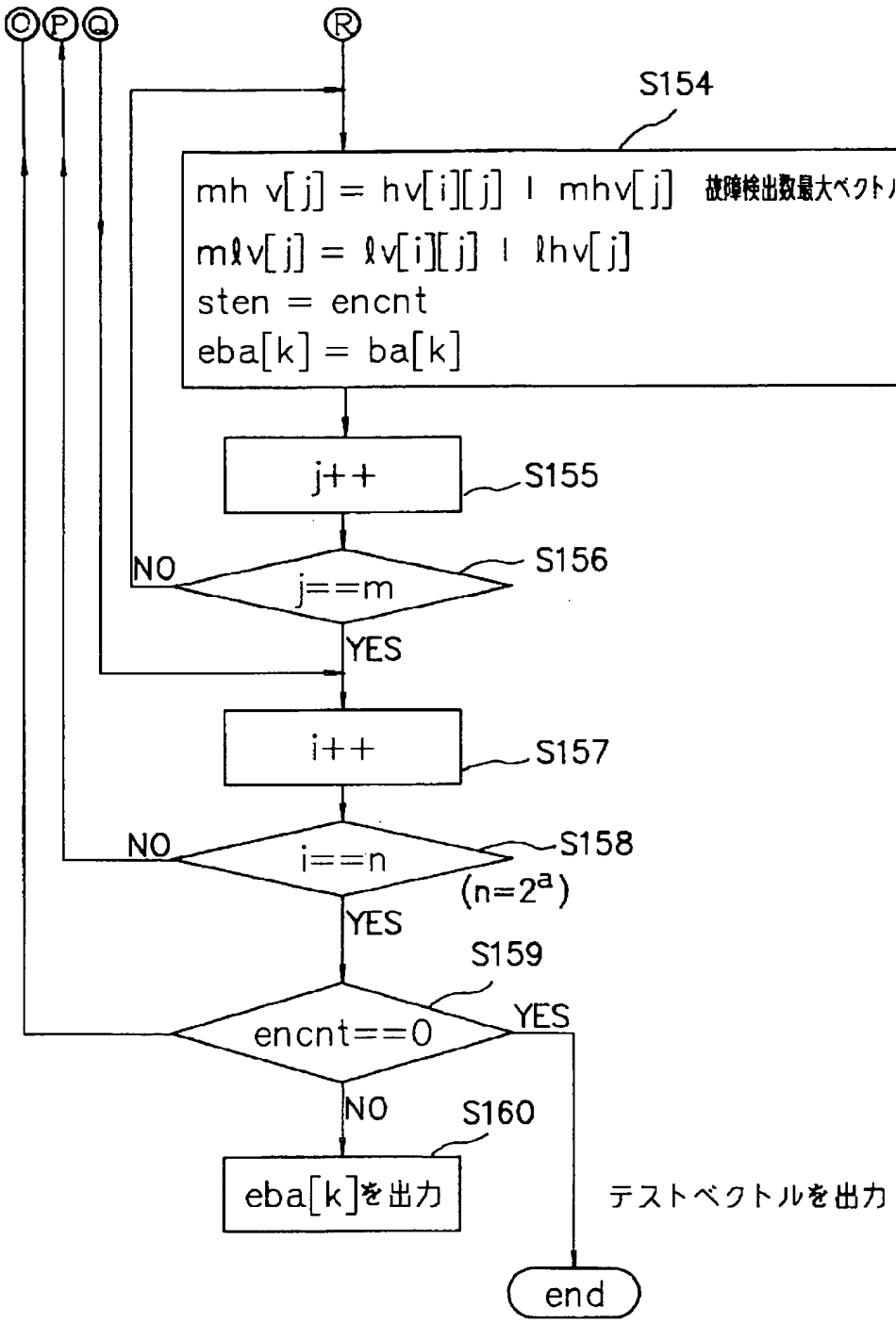
【図14】


L
K
S135
st[m] = 1
st[m−1] = 1
st[m−b+1] = 1
出力端子数bのとき
st[z] = st[ottm[z]] & funst(intm[ottm[z])
st[0] = st[ottm[0]] & funst(intm[ottm[0]])
j = 0
S136
S137
hv[i][j] = st[j] & (not ac[j])
入力ベクトルがiの時のstack at high検出可能なノード
lv[i][j] = st[j] & ac[j]
入力ベクトルがiの時のstack at low検出可能なノード
j++
S138
S139
NO
j == m
YES
stcnt = "hv[i]のHの数" +"lv[i]のHの数"
S140
NO
stcnt > stmax
S141
YES
L
M
N

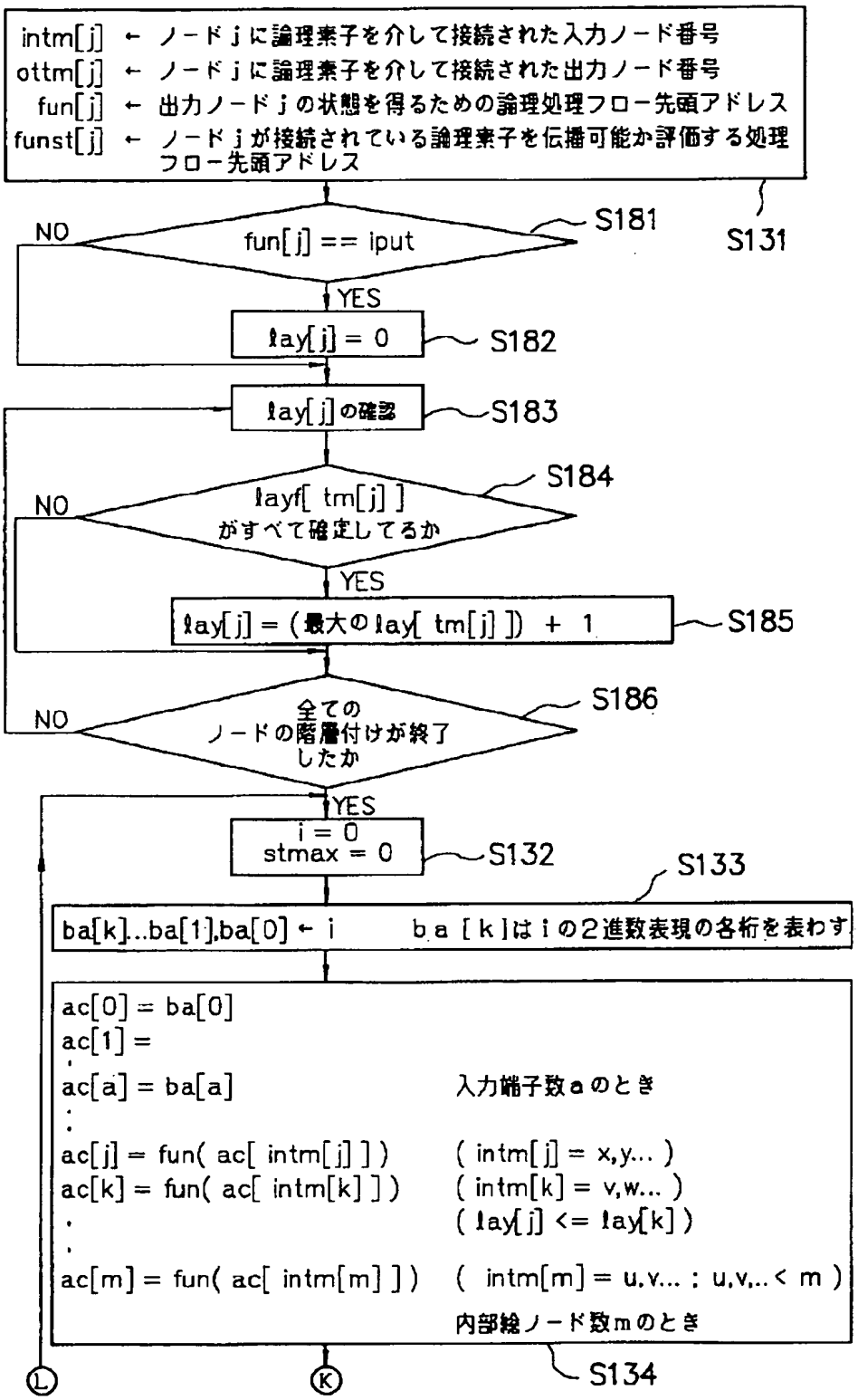
【図15】


L
M
N
j = 0
S142
mhv[j] = hv[i][j] 故障検出数最大ベクトル探索
mlv[j] = lv[i][j]
stmax =stcnt
mba[k] = ba[k]
S143
j++
S144
NO
j=m
S145
YES
i++
S146
S147
NO
i == n
(n=2^a)
YES
S148
mba[k] を出力
テストベクトルを出力
S149
enmax = 0
前述 ② 処理スタート
i = 0
encnt = 0
S150
S151
encnt = "((mhv[j] == 0) & hv[i][j]) + ((mlv[j] == 0) & lv[i][j])" の1の数
NO
encnt > enmax
S152
YES
j = 0
S153
O
P
Q
R

【図16】


O
P
Q
R
S154
mh v[j] = hv[i][j] | mhv[j] 故障検出数最大ベクトル探索
mℓv[j] = ℓv[i][j] | ℓhv[j]
sten = encnt
eba[k] = ba[k]
j++
S155
NO
j==m
S156
YES
i++
S157
NO
i==n
S158
(n=2^a)
YES
S159
encnt==0
YES
NO
S160
eba[k]を出力
テストベクトルを出力
end

【図18】


intm[j] ← ノードjに論理素子を介して接続された入力ノード番号
ottm[j] ← ノードjに論理素子を介して接続された出力ノード番号
fun[j] ← 出力ノードjの状態を得るための論理処理フロー先頭アドレス
funst[j] ← ノードjが接続されている論理素子を伝播可能か評価する処理フロー先頭アドレス
S131
fun[j] == iput
NO
YES
S181
lay[j] = 0
S182
lay[j]の確認
S183
layf[tm[j]] がすべて確定してるか
S184
lay[j] = (最大のlay[tm[j]]) + 1
S185
全てのノードの階層付けが終了したか
S186
i = 0
stmax = 0
S132
S133
ba[k]..ba[1],ba[0] ← i
b a [k]はｉの2進数表現の各桁を表わす
ac[0] = ba[0]
ac[1] =
ac[a] = ba[a]
入力端子数aのとき
ac[j] = fun(ac[intm[j]])
(intm[j] = x,y...)
ac[k] = fun(ac[intm[k]])
(intm[k] = v,w...)
(lay[j] <= lay[k])
ac[m] = fun(ac[intm[m]])
(intm[m] = u,v... : u,v,... < m)
内部線ノード数mのとき
S134
L
K